OULUN AMMATTIKORKEAKOULU

Mika Manninen

DIGITAALINEN KOSTEUSTARKASTUSLOMAKE KIINTEISTÖHUOLTOYRITYKSELLE

DIGITAALINEN KOSTEUSTARKASTUSLOMAKE KIINTEISTÖHUOLTOYRITYKSELLE

Mika Manninen
Opinnäytetyö
Kevät 2018
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma, Web-sovelluskehitys

Tekijä: Mika Manninen
Opinnäytetyön nimi: Digitaalinen kosteustarkastuslomake kiinteistöhuoltoyritykselle
Työn ohjaaja: Teppo Räisänen
Työn valmistumislukukausi ja -vuosi: Kevät 2018 Sivumäärä: 26

TaloPlus Oy on kuntotarkastuksia tekevä oululainen kiinteistöhuoltoyritys. Opinnäytetyön tarkoituksena oli kehittää kosteusvaurioiden tarkastamisessa käytettävästä paperilomakkeessa digitaalinen versio. Koska kosteustarkastuksia tehdään asiakkaan tiloissa, oli lähtökohtana kehittää järjestelmästä mobiililaitteilla käytettävä. TaloPlus Oy:n toiminnan kannalta digitalisoitu mobiiliratkaisu on hyvä, koska se nopeuttaa tarkastusten tekoa sekä vähentää virheitä.

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa mobiililaitteilla toimiva Web-sovellus, joka täyttää toimeksiantajan vaatimukset. Toimeksiantaja haluaa muun muassa käyttää sovellusta tarkistuskohteessa ja sen tulee automatisoida tarkistusten raportoimista ja tallentamista. Toimeksiannon aluksi kartoitettiin TaloPlus Oy:n kosteustarkastusprosessin alkutilanne ja määriteltiin millaisia ominaisuuksia yritys haluaisi uuteen Web-sovellukseen. Tämän pohjalta suunniteltiin ratkaisu yrityksen tarpeisiin.

Opinnäytetyön lopputuloksena syntyi mobiililaitteilla käytettävä Web-sovellus, jonka asiakas voi ottaa käyttöönsä, kunhan he saavat Web-palvelimen toimimaan. Raportissa pääpaino on sovelluksen suunnittelussa ja toteutuksessa. Sovellus toteutettiin käyttäen HTML-, CSS-, Bootstrap-, PHP- ja PHPWord -tekniikoita.

Tämän opinnäytetyön teoriaosuudessa kuvataan läpi sovelluksen ohjelmointikielet ja niiden taustaa. Asiakas on lopputulokseen tyytyväinen.

Asiasanat: Kuntotarkastus, websovellus, HTML, CSS, Bootstrap, PHPWord

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Web Application Development

---

Author: Mika Manninen
Title of thesis: Digital humidity control form for housing company
Supervisor: Teppo Räisänen
Term and year when the thesis was submitted: Spring 2018 Number of pages: 26

The commissioner for this thesis was Oulu based company Taloplus Oy. Subject of this thesis was to develop a digital version of company's current paper-based solution. Since the commissioner is doing maintenance in customer locations, it's vital that the solution can be used on mobile devices. For TaloPlus Oy digitalizing mobile solution is great because it speeds up the process of making reports and reduces the number of mistakes.

The aim of this thesis was to design and develop a web application that can be used on mobile devices. Commissioner wants to use this application in maintenance check locations and therefore the application should also automatically create and save the reports.

The result of this thesis includes a mobile application that can be used as soon as the commissioner sets up a web server. Programming languages which were used to make this application are HTML, CSS, Bootstrap, PHP and PHPWord.

The theory section describes the programming languages that were used in this thesis. Commissioner was pleased with the result.

Keywords: Maintenance, Web application, HTML, CSS, Bootstrap, PHPWord

SISÄLLYS

1 JOHDANTO

Yhteiskunnallinen muutos kohti globaalia palveluyhteiskuntaa on pakottanut yritykset ajattelemaan tarkemmin sitä, millä tavalla yritykset tarjoavat palveluja asiakkailleen. Enää ei riitä, että asiakkaille tarjotaan peruspalvelua, vaan asiakkaat haluavat ja vaativat enemmän. Työ- ja elinkeinoministeriön (2015, viitattu 22.3.2018) julkaisussa painotetaan sitä, kuinka teollisuuden tulisi rakentaa kilpailuetua luomalla uusia innovatiivisia ratkaisuja. Digitalisaatio lisää globaalia kilpailua arvoverkostoissa, ja palvelujen toimintatavan muutoksen vuoksi kansainvälinen kilpailu haastaa kotimaiset palvelun-tarjoajat. On olennaista, että yritykset kiinnittävät huomiota kannustavaan toimintaympäristöön ja keskittyvät vahvistamaan tuotekehitystä ja tutkimusta.

Digitaalisten laitteiden käyttö on jatkanut kasvua 2010-luvulla (Suomen virallinen tilasto, viitattu 22.3.2018). Erityisesti mobiililaitteiden käyttö on yleistynyt enemmän verrattuna tietokoneisiin, ja niiden teho on myös huomattavasti kasvanut. Taskuun mahtuvan älypuhelimen prosessointiteho on jo lähellä kannettavia tietokoneita. Tämä avaa mahdollisuuksia uusille ratkaisuille, jotka eivät olleet ennen mahdollisia, koska tietokoneiden käyttö ulkona ja pienissä tiloissa ei ole käyttäjälle käytännöllistä. Rakennusalalla voi olla erityisen tärkeää, että on mahdollisuus käyttää laitetta ulkona. Koska kannettavat ovat harvemmin vedenkestäviä, se avaa uusia mahdollisuuksia älypuhelimille.

Mobiililaitteiden yleistyminen ja internetin selaaminen mobiilissa on johtanut siihen, että kehittäjät käyttävät enemmän aikaa siihen, miten sovellukset näkyvät ja toimivat mobiililaitteilla. Kehitystyö lähtee siitä, että ensiksi suunnitellaan sovellus kännykälle sopivaksi ja sen jälkeen mietitään, miltä se näyttää pöytäkoneella. Tätä toimenpidettä kutsutaan ”mobile first” -lähestymistavaksi.

Tässä opinnäytetyössä keskitytään TaloPlus Oy:ltä tulleeseen toimeksiantoon. TaloPlus on kunto-tarkastuksia tekevä oululainen kiinteistöhuoltoyritys. Opinnäytetyön tarkoituksena oli kehittää kosteusvaurioiden tarkastamisessa käytettävästä paperilomakkeesta digitaalinen versio. Koska kosteustarkastuksia tehdään asiakkaan tiloissa, lähtökohtana oli kehittää järjestelmästä mobiililaitteilla käytettävä. Toimeksiantaja oli jo aloittanut siirtymisen digitaaliseen aikaan ja hankkinut Samsungin tabletteja, joita on tarkoitus käyttää työkohteissa.

TaloPlus Oy:n toiminnan kannalta digitalisoitu mobiiliratkaisu olisi hyvä. Järjestelmä nopeuttaa tarkastusten tekoa sekä vähentää virheitä. Tarkastajan on helppo tehdä tarkastukseen liittyvät toimenpiteet esimerkiksi tablet-laitteella ja muokata tarkastusraporttia myöhemmin toimistolla pöytäkoneella. Valmiin raportin voi tulostaa ja arkistoida sekä näyttää asiakkaalle tarvittaessa.

Opinnäytetyön rakenne on seuraava. Luvussa 2 esitellään kehittämistehtävän toteutuksessa käytetyt ohjelmointikielet. Luvussa 3 esitellään kehitetyn mobiilisovelluksen suunnittelu ja toteutus. Luvussa 4 on opinnäytetyön pohdinta.

2 KEHITTÄMISTEHTÄVÄSSÄ KÄYTETYT OHJELMOINTIKIELET

Tässä osiossa käydään läpi opinnäytetyössä käytettyjä tekniikoita sekä merkkaus- ja ohjelmointikieliä. Näitä ovat HTML, CSS, PHP, Bootstrap ja PHPWord.

2.1 HTML

Modernit web-sovellukset perustuvat HTML-, CSS-, PHP- ja JS-kieliin. Tässä sovelluksessa käytin kaikkia muita paitsi Javascript-kieltä (JS), koska sille ei ollut tarvetta tässä sovelluksessa. HTML- ja CSS -kielillä luotiin sovelluksen käyttöliittymä, koska ne ovat yleisimpiä merkintäkieliä, joita käytetään web-sovellusten tekemiseen. PHP toimii sovelluksen palvelimena ja tiedostojen välittäjänä.

HTML on lyhennys sanoista Hypertext Markup Language. Se tunnetaan yleisesti sinä kielenä, jolla verkkosivustot on kirjoitettu. Sen kehittäminen alkoi vuonna 1989, kun World Wide Webin keksijänä tunnettu Tim Berners-Lee alkoi kehittämään järjestelmää, jonka avulla CERNin tutkijat eri puolilla maailmaa voisivat kerätä ja jakaa tietoa keskenään verkon välityksellä. Järjestelmän tarkoitus ei ollut tarjota isoa määrää dokumentteja, joita voitaisiin ladata yksittäisille käyttäjille, vaan se mahdollistaisi tekstien linkittämisen suoraan tiedostoihin. (Addison Wesley Longman 1998, viitattu 10.1.2018.)

HTML-merkintäkieli koostuu tekstistä, joka muodostuu eri elementeistä. Useimmiten näissä elementeissä on pareittain ensin aloitustunniste ja lopuksi lopetustunniste, joiden väliin tulee sisältö, mikäli elementti tarvitsee sisällön. On myös elementtejä, jotka eivät toimi pareittain vaan ne itse toimivat elementin aloittajana ja lopettajana, kuten elementti img. (Aas ym. 2018, viitattu 25.2.2018)

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;charset=UTF-8">
  <title>Tämä on sivun otsikko</title>
</head>
<body>
  <h1>Hello World!</h1>
  <p>Tämä on kappale</p>
</body>
</html>
```

KUVIO 1. Esimerkki HTML-sivun koodista käyttäen tunnettua ”Hello World”-esimerkkiä

Kuviossa 1 on esitelty yksinkertainen esimerkki HTML-dokumentista. HTML-dokumentissa tulee olla <html>, <head> ja <body>–tagit. <html>-tag kertoo selaimelle tiedoston olevan HTML-merkkikieltä, jolloin selain osaa tulkita koodia oikein. <head>-tag sisältää dokumentin nimen ja mahdollisesti määritellyn tyylitiedoston. <body>-tagiin on kirjoitettu dokumentin sisältö.

Kuten kuviosta 1 huomaa, HTML -kielessä käytetään alku- ja lopputageja, joiden väliin tulee sisältö. Tagit <title>, <h1> ja <p> ovat tästä yksinkertaisimmat esimerkit. On myös olemassa elementtejä, joissa on itsessään alku- ja lopputagi, esimerkiksi kuvan esittävä tagi <img>, joka kirjoitetaan: .

2.2 Cascading Style Sheets

Cascading Style Sheets (CSS) on merkkauskieli, joka on suunniteltu mahdollistamaan HTML-dokumenttien visuaalisen muokkaamisen. Sen avulla voi määritellä esimerkiksi fontin koon, taustavärin, taustakuvan, taulukkojen tyylin tai elementtien sijainnin sivulla. CSS:n yksi isoimmista eduista on se, että sillä voi muokata useita dokumentteja kerralla, jolloin ei tarvitse määrittää jokaisen dokumentin ulkoasua erikseen. (Pouncey & York 2011, 3.)

Ensimmäisinä vuosina, jolloin internet oli luotu, käyttäjät eivät pystyneet vaikuttamaan siihen, miltä heidän tekemänsä sivut näyttivät. Koska internet oli luotu tieteellistä tarkoitusta varten, käyttäjiä kiinnosti enemmän dokumenttien sisältö kuin se, miltä ne näyttivät. Tästä syystä dokumenttien ulkonäköön ei panostettu. Kun internetiä alettiin käyttää tieteellisten piirien ulkopuolella, se toi samalla uusia käyttäjiä, joita kiinnosti se, miltä sivustot näyttivät. Näillä käyttäjillä oli usein kokemusta paperipohjaisista julkaisuympäristöistä, joissa heillä oli esityksen täysi hallinta. He halusivat vaihtaa

tekstin väriä, saada sen näyttämään tiheämmältä tai siirtää sen mihin itse halusivat. (Håkon & Bert 2005, 3.)

Vuonna 1994 Håkon julkaisi ensimmäisen dokumentin, jossa hän kuvaili, miten dokumenttien visuaalisuutta voitaisiin muokata. Hän kutsui tätä tapaa porrastetuiksi tyyliarkeiksi, joka tunnetaan paremmin lyhenteellä CSS. Håkon ei ollut ainoa, joka oli alkanut kehittämään HTML-dokumenttien visuaalista ilmettä muokkaavaa kieltä. Pei Wei oli kehittänyt merkkauskieltä Viola-selaimelle, ja Robert Raisch oli kehittänyt samankaltaista kieltä kuin Håkon. Håkonin tekemä CSS oli parempi, koska se otti huomioon lukijan, tekijän ja laitteen vaatimukset. (Håkon & Bert 1999, 4.)

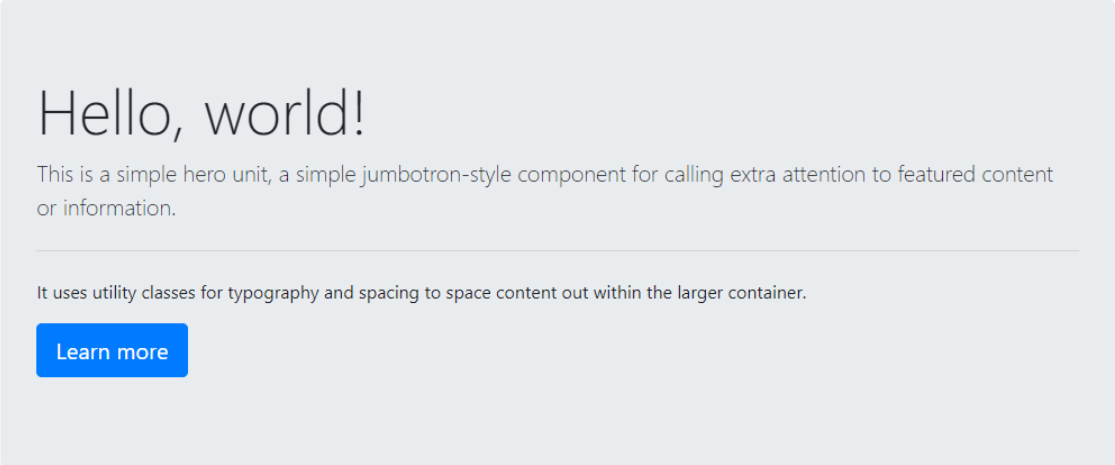
```
1  body {
2    background-color: beige;
3  }
4  p {
5    font-size: 20px;
6  }
7
```

KUVIO 2. Esimerkki CSS-luokista

Kuviossa 2 on esitelty esimerkki CSS-luokista. CSS toimii siten, että ensiksi valitaan elementti, ja sen jälkeen määritellään elementille haluttuja arvoja. Kuvion 2 tiedostossa HTML-dokumentille annetaan beige taustaväri ja kaikille tekstikappaleille annetaan fonttikooksi 20 pikseliä.

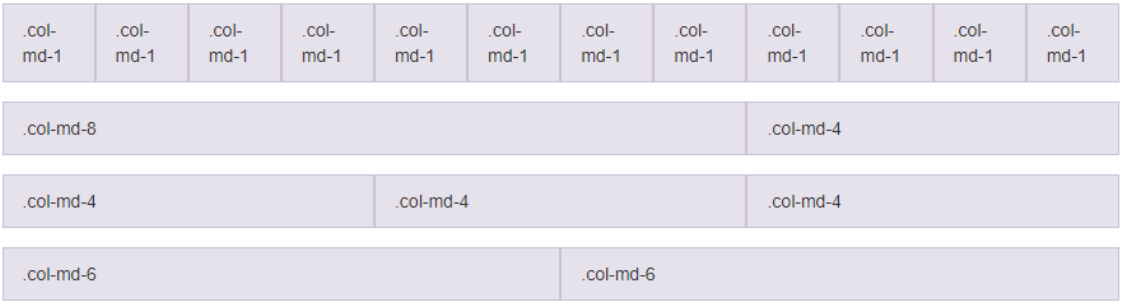
2.3 Bootstrap

Bootstrap on avoimeen lähdekoodiin perustuva kirjasto, jossa on valmiiksi luokiteltu eri CSS-luokkia verkkosivustojen luomisen helpottamiseksi (Bootstrap 2018a). Sen avulla kehittäjä voi nopeasti aloittaa verkkosivuston kehittämisen ilman, että hän joutuu kirjoittamaan responsiivisen ulkoasun alusta alkaen itse.



KUVIO 3. Bootstrap jumbotron -elementti (Bootstrap 2018c, viitattu 22.3.2018)

Bootstrap sisältää valmiita tyylejä erilaisille Web-elementeille, mikä nopeuttaa huomattavasti kehitystyötä, koska kehittäjä voi kiinnittää huomion ominaisuuksien tekemiseen ja vähemmän ulkoasun rakentamiseen. Se sisältää erilaisia valmiiksi määriteltyjä elementtejä Web-suunnittelua varten, kuten esimerkiksi ”Card” tai ”Jumbotron”. Jumbotronin avulla (kuvio 3) luodaan elementti, jonka avulla voidaan esitellä tuotteen tai yrityksen palvelu selkeästi.



KUVIO 4. Bootstrap ruudukkojärjestelmä (Bootstrap 2018b, viitattu 1.3.2018)

Bootstrap toimii kaikilla mobiili- ja verkkoselaimilla. Vanhemmat selaimet voivat näyttää Bootstrapin eri tavalla, mutta toimivat kuitenkin hyvin. Bootstrap tarjoaa monia tapoja muokata sen eri osia omanlaiseksi. Kehittäjä voi esimerkiksi muokata helposti kirjaston värejä haluamakseen ja jättää käyttämättä jotain osaa, kuten ruudukkoa. Kuviossa 4 on esimerkki Bootstrapin tekemästä ruudukkojärjestelmästä. Sen avulla kehittäjä voi luoda ketterästi responsiivisen käyttöliittymän, joka toimii kaikilla eri näyttölaitteilla. Ruudukkojärjestelmä perustuu 12 laatikon muodostamaan ruudukkoon. Bootstrapin huonona puolena voidaan nähdä se, että muokkausominaisuuksista huolimatta sivustot, jotka on tehty Bootstrapilla, näyttävät usein samanlaisilta. (Niska 2014, 7.)

2.4 PHP

PHP on palvelimella toimiva ohjelmointikieli. Kun käyttäjä vierailee PHP-sivustolla, palvelin tunnistaa dokumentissa olevan PHP-koodin ja tekee siinä olevat toiminnot, kuten laskee laskuja tai kirjoittaa tietyn tekstikappaleen. Tämän jälkeen palvelin palauttaa Web-palvelimelle HTML-dokumentin, joka lähetetään selaimelle, jonka käyttäjä näkee näytöstä. (Meloni & Premier 2005, 2.)

PHP:n pääkehittäjäksi voidaan luonnehtia Rasmus Lerdorf, ja vuonna 1994 kehitti uudelleenkäytettäviä ratkaisuja websivujen suunnittelun ja toteutuksen tueksi. Näihin lukeutui muun muassa, vieraskirja, laskuri ja muita kotisivuelementtejä. Nämä elementit olivat innovatiivisia ja uusia silloinkin, kun internet oli vasta alkuvaiheessa (Meloni & Premier 2005, 5.)

Dynaamisesti luotujen verkkosivustojen tekeminen on helppoa PHP:n avulla. PHP-kieltä voidaan lisätä HTML-dokumenttien sekaan, joten sen käyttäminen olemassa olevien dokumenttien kanssa on helppoa. Se eroaa muista kielistä, kuten Javascriptistä, koska koodi suoritetaan palvelimella, joka sitten lähetetään käyttäjälle. (Achour, Betz, Dovgal & Lopes 2018.)

```
6  <body>
7    <h1>Hello World!</h1>
8    <?php
9      $x = 5;
10     $y = 4;
11     echo "<p>Tämä kappale tulee PHP-palvelimelta</p>";
12     echo $x . " + " . $y . " = " . ($x + $y);
13   ?>
14   <p>Tämä on kappale</p>
15 </body>
```

KUVIO 5. Esimerkki PHP-koodista

PHP-koodia voidaan kirjoittaa HTML-koodin väliin, ja PHP-palvelin lukee sen ennen kuin palauttaa sen käyttäjälle HTML-dokumenttina. Kuviossa 5 olevassa koodissa PHP-palvelin tulostaa erillisen kappaleen tekstiä. Tämän kappaleen alle palvelin tulostaa muuttujat x ja y ja niiden summan. Eri muuttujia ja tekstiä voidaan yhdistää, mutta ne pitää erottaa pisteillä, jotta palvelin tunnistaa ne eri elementeiksi.

2.5 PHPWord

PHPWord on PHP-kielellä kirjoitettu avoimen lähdekoodin projekti, joka mahdollistaa erilaisten dokumenttiformaattien lukemisen ja kirjoittamisen. Se sisältää valmiita PHP-luokkia, joiden avulla kehittäjä voi kirjoittaa, tallentaa ja muokata Open XML-, ODF- ja RTF-dokumenttityyppejä. Sen muihin ominaisuuksiin sisältyy muun muassa mahdollisuus muokata otsikkoa, dokumentin tekijää, lisätä taulukoita, muokata ylä- ja alatunnistetta ja luoda dokumentteja malleista. (PHPWorda 2017.)

```php
<?php
require_once 'bootstrap.php';

// Creating the new document...
$phpWord = new \PhpOffice\PhpWord\PhpWord();

/* Note: any element you append to a document must reside inside of a Section. */

// Adding an empty Section to the document...
$section = $phpWord->addSection();
// Adding Text element to the Section having font styled by default...
$section->addText(
    '"Learn from yesterday, live for today, hope for tomorrow. '
        . 'The important thing is not to stop questioning." '
        . '(Albert Einstein)'
);
```

Kuvio 6. Esimerkki PHPWord koodista (PHPWordb 2017, viitattu 21.3.2018)

PHPWordia käytetään luomalla ensin luokka ja sen jälkeen luomalla kappale, jota voi muokata vapaasti esimerkiksi lisäämällä tekstiä addText-funktiolla (kuvio 6) tai muokkaamalla ylätunnistetta addHeader-funktiolla. Tässä työssä PHPWordia käytettiin tuottamaan Word-dokumentti lomakkeelle annetuista tiedoista ja tallentamaan se palvelimelle myöhempää käyttöä varten.

3 TOIMEKSIANTO JA TOTEUTUS

Tässä luvussa esitellään toimeksiantaja, toimeksianto, alkutilanne ja sovelluksen toteuttaminen.

3.1 Toimeksiantajan esittely

Taloplus on vuonna 2008 perustettu suomalainen yritys, joka on erikoistunut erilaisiin rakennusten kuntotarkastuksiin ja rakennusten laadunvarmistukseen. Yrityksen palveluihin kuuluvat mm. kuntotarkastukset, kosteusmittaukset, lämpökuvaukset, tiiviysmittaukset ja energiatodistukset. Taloplus toimii Oulun seudulla ja tarvittaessa muualla Suomessa. Toiminta perustuu pitkään kokemukseen rakennusalalla ja ammattitaitoiseen henkilökuntaan. Yrityksen asiakkaisiin lukeutuvat mm. rakennusliikkeet, rakennuttajat, yksityishenkilöt, isännöitsijät ja asiantuntijat (Taloplus 2018, viitattu 22.3.2018).

Yritys otti yhteyttä Oulun ammattikorkeakoulun Informaatioteknologian tietojenkäsittelyn tiimiin. Se halusi korvata nykyisen paperisen lomakkeen sähköisellä ratkaisulla. Löysin toimeksiannon kouluni ilmoitustaulu Oivasta. Otin yhteyttä yritykseen sähköpostitse ja kerroin olevani kiinnostunut projektista. Asia eteni toimeksiantoon.

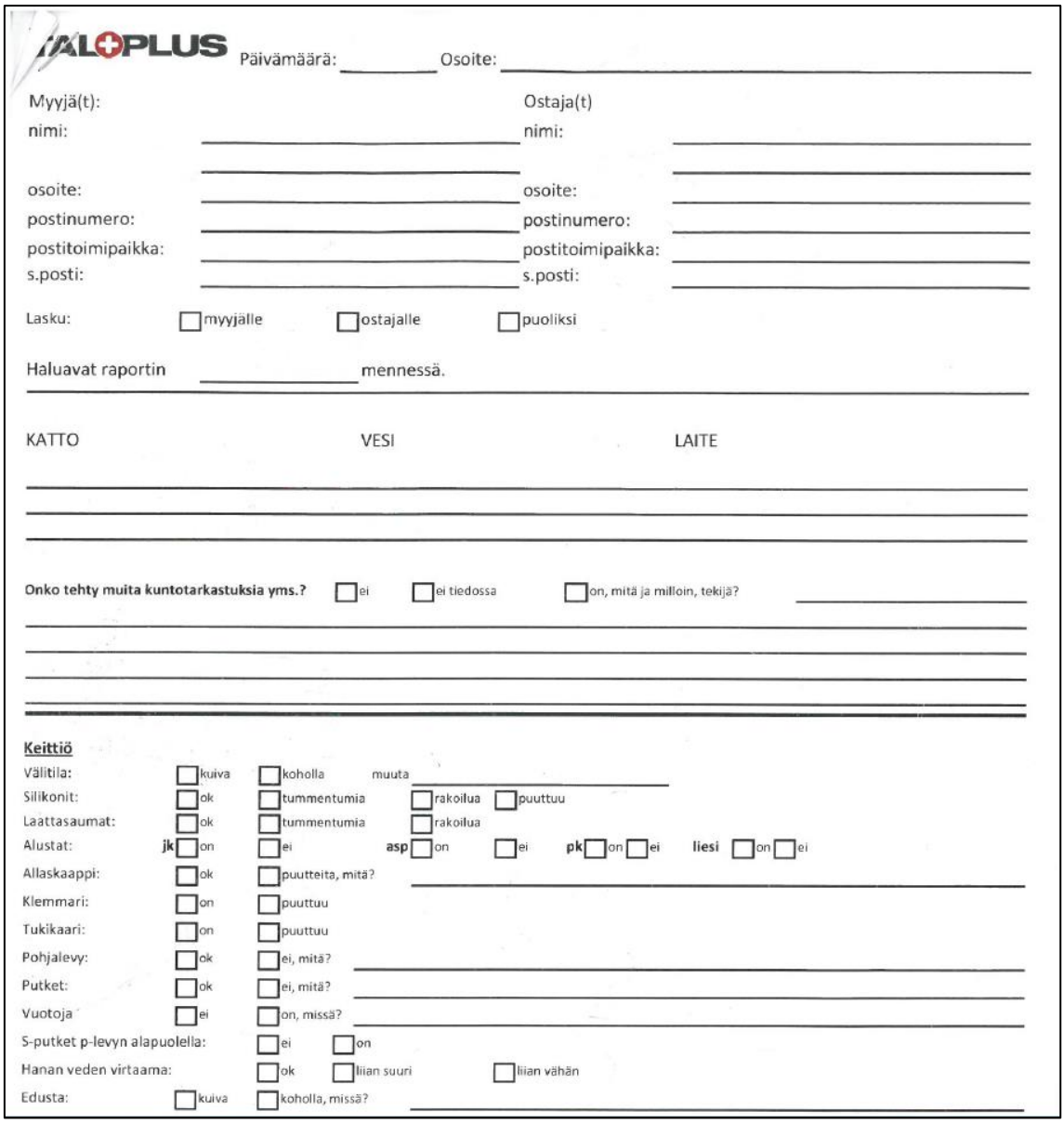
3.2 Toimeksiantajan tapaaminen

Kävin tapaamassa yrityksen edustajia heidän toimitiloissaan ensimmäisen kerran 18.10.2016. Tapaamisessa tarkastelimme tämänhetkistä tilannetta ja sitä, miten he haluaisivat parantaa sitä. Sain mukaani tuolloin käytetyt lomakkeet, jotta voin niiden avulla saada paremman käsityksen kosteusmittauksista. Yritys antoi minulle vapaat kädet toteutukseen. Tässä vaiheessa aloin suunnitella sovelluksen rakennetta ja mahdollista toteutustapaa.

Seuraavassa tapaamisessa 14.2.2017 esittelin aikaansaannoksiani, kuten käyttöliittymän luonnosta ja alustavaa sovelluksen rakennetta. Kävimme läpi myös nykyiseen lomakkeeseen tulevia muutoksia, kuten erilaisia tekstikenttien muutoksia ja joidenkin kenttien kokonaan poistamista. Jatkoin sovelluksen suunnittelua yrityksen toiveiden mukaisesti. Yritys toivoi sovelluksen olevan helppokäyttöinen ja yksinkertainen.

Esittelin yritykselle seuraavan kerran työni edistymistä 3.7.2017. Työssä olin edennyt jo sovelluksen käyttöliittymän toteuttamiseen. Kävimme läpi sovelluksen käyttöliittymää ja sitä, mitä mieltä yritys oli siitä. Sain hyvää palautetta, ja sovimme muutamasta muutoksesta. Muutoksiin lukeutui muun muassa muutoksia fontin kokoon ja tekstikenttien suurentamiseen. Sovellusta on jatkuvasti kehitetty yrityksen toiveet huomioiden, ja näin yritys saa juuri haluamansa sovelluksen käyttöönsä.

Tällä hetkellä, kun asiakas on tilannut kosteusmittauksen, yritys lähettää kosteustarkastajan, joka käy läpi kohteen huone kerrallaan. Kosteustarkastaja käyttää työkalunaan kosteusmittaria ja raportoi työn tuloksen 5-sivuiselle paperilomakkeelle. Lomakkeen sivut on yksilöity sijaintien mukaan. Esimerkiksi keittiössä on erilaiset tarkistuskohteet kuin saunassa. Lomake on pitkä ja aikaa vievä. Mittauksen tulokset tulee merkitä kirjoittamalla. Vastauskohtia lomakkeessa on monia. Kun lomake on täytetty ja kohde on tarkistettu, tarkastaja palaa toimistolle ja kirjoittaa Word-raportin asiakkaalle. Tarkastaja kirjoittaa samat asiat kahteen kertaan. Ensin tarkastuskohteessa lomakkeelle ja sen jälkeen toimistolla raporttiin. Tässä raportissa tarkastaja kuvailee eri kohtia ja huomauttaa, mikäli kohteesta löytyy kosteusvaurioita tai muuta huomautettavaa. (Tanninen, haastattelu 18.10.2016.)



TALOPLUS Päivämäärä: _____ Osoite: _____

Myyjä(t): Ostaja(t)

nimi: _____ nimi: _____

_____ _____

osoite: _____ osoite: _____

postinumero: _____ postinumero: _____

postitoimipaikka: _____ postitoimipaikka: _____

s.posti: _____ s.posti: _____

Lasku: ☐ myyjälle ☐ ostajalle ☐ puoliksi

Haluavat raportin _____ mennessä.

KATTO VESI LAITE

Onko tehty muita kuntotarkastuksia yms.? ☐ ei ☐ ei tiedossa ☐ on, mitä ja milloin, tekijä? _____

Keittiö

Välitila: ☐ kuiva ☐ kohollaa muuta _____

Silikonit: ☐ ok ☐ tummentumia ☐ rakoilua ☐ puuttuu

Laattasaumat: ☐ ok ☐ tummentumia ☐ rakoilua

Alustat: jk ☐ on ☐ ei asp ☐ on ☐ ei pk ☐ on ☐ ei liesi ☐ on ☐ ei

Allaskaappi: ☐ ok ☐ puutteita, mitä? _____

Klemmari: ☐ on ☐ puuttuu

Tukikaari: ☐ on ☐ puuttuu

Pohjalevy: ☐ ok ☐ ei, mitä? _____

Putket: ☐ ok ☐ ei, mitä? _____

Vuotoja ☐ ei ☐ on, missä? _____

S-putket p-levyn alapuolella: ☐ ei ☐ on

Hanan veden virtaama: ☐ ok ☐ liian suuri ☐ liian vähän

Edusta: ☐ kuiva ☐ koholla, missä? _____

KUVIO 7. Nykyinen lomake

Kuviossa 7 on esitelty olemassa oleva paperinen tarkastuslomake. Lomakkeen alussa on tarkastuskohteen perustiedot. Tämän jälkeen lomakkeessa kysytään tarkempia tietoja kohteesta ja sen kunnosta. Lomakkeen alaosassa on tarkempia kysymyksiä keittiön kuntoon liittyen.

3.3 Sovelluksen suunnittelu

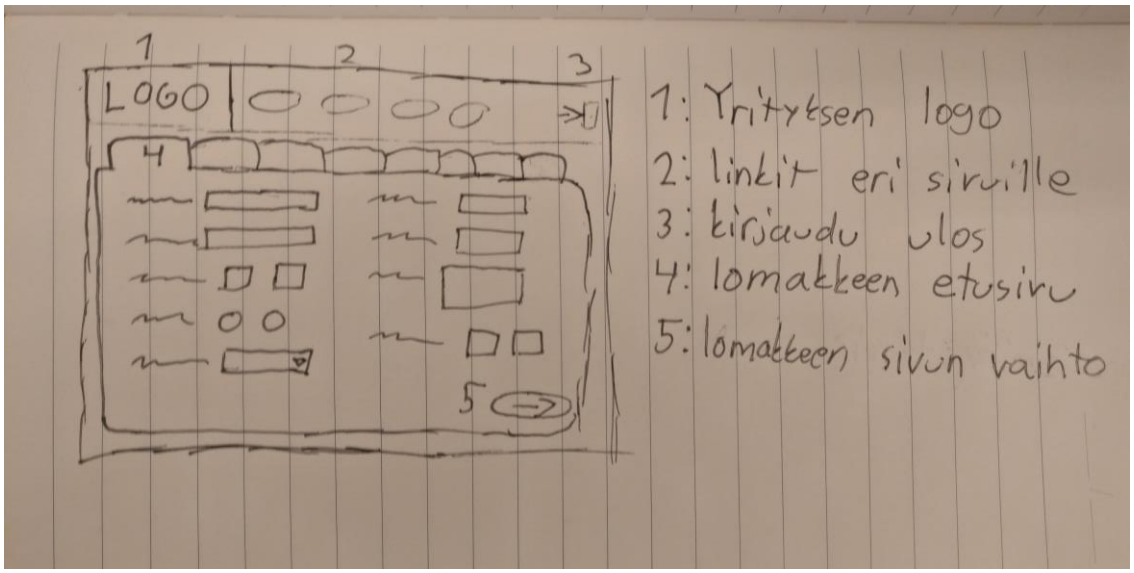
Ennen kuin aloitin toteuttamaan sovellusta, minun oli valittava, mitä teknologioita tulen käyttämään. Aloitin pohtimalla sovelluksen arkkitehtuuria ja sovelluksen palvelinta. Mietin myös, miten tieto ete-

nee, kun se on tallennettu järjestelmään. Ensimmäinen vaihtoehto oli tehdä suoraan Wordilla lomake, jonka voi täyttää tabletilla missä tahansa. Tämä ratkaisu olisi toiminut melkein samalla tavalla kuin nykyinen ratkaisu. Ero olisi ollut siinä, että lomake olisi ollut sähköisessä muodossa. Etuna olisi ollut, että näin ei olisi vaadittu käyttäjäjärjestelmää ja lomake olisi aina yrityksen omilla tableteilla ja ympäristössä. Huono puoli olisi se, että tämä ei toisi ratkaisua yrityksen ongelmaan.

Toinen vaihtoehto olisi rakentaa sovellus perustumaan tietokantapohjaiseksi. Sovellus ensin tallentaisi tiedot HTML-lomakkeelta Mongodb-tietokantaan. Kun käyttäjä painaa halutun dokumentin linkkiä, sovellus hakee kyseisen dokumentin tiedot tietokannasta, muuntaa sen Word-dokumentiksi ja lähettää sen käyttäjälle. Tämä ratkaisu osoittautui liian haastavaksi ohjelmoida, ja en ehkä olisi kyennyt tekemään sovellusta tällä arkkitehtuurilla ja aikataululla. Ongelmaksi nousi se, että olisi pitänyt opetella uusi palvelinratkaisu sovellukselle, joka perustuu NodeJS-kieleen, ja se olisi vienyt liikaa aikaa itse suunnittelusta ja toteutuksesta.

Lopulta päädyin ratkaisuun, jossa sovellus luo Word-tiedostot HTML-lomakkeesta automaattisesti ja tallentaa ne PHP-palvelimelle. Tutkimalla eri ohjelmointikirjastoja löysin PHPWord-kirjaston, jonka käyttäminen oli selkeää ja yksinkertaista. Kun olin tutustunut PHPWordin dokumentointiin, näin sen olevan paras ratkaisu toteuttaa sovellus. Tämä ratkaisu oli hyvä myös siksi, koska minulla oli jo ennestään kokemusta PHP-kielestä ja näin pystyin priorisoimaan ajankäyttöni muihin tehtäviin, kuten suunnitteluun ja toteutukseen. Kun esittelin ratkaisuni yritykselle, heille sopi tämän ratkaisun arkkitehtuuri.

Tämän jälkeen aloin suunnittelemaan sovelluksen käyttöliittymän toteuttamista. Suunnittelemisen aloitin piirtämällä mahdollisen sovelluksen käyttöliittymän paperille. Tein muutamia luonnoksia sovelluksen käyttöliittymästä ja arvioin niiden hyviä ja huonoja puolia. Näiden luonnoksien arvostelujen perusteella päätin, mitä luonnosta lähdin toteuttamaan itse sovellukseksi. Näin jälkeenpäin mietittynä, olisi ollut hyvä näyttää luonnoksia yrityksen työntekijöille ja pyytää heiltä palautetta.

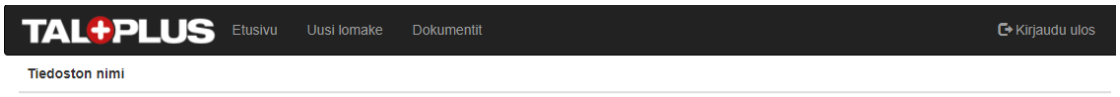


KUVIO 8. Luonnos käyttöliittymästä

Kuviossa 8 ylimmällä rivillä on sovelluksen navigointi, eli sitä kautta voi siirtyä sovelluksen eri sivuille, kuten etusivulle. Ylärivillä on myös yrityksen logo ja uloskirjautuminen. Logon sijainti on mietitty tarkkaan, jotta visuaalinen tyyli on mahdollisimman näyttävä. Logo on näkyvällä paikalla, jotta käyttäjä tietää olevansa oikealla sivulla. Painike, josta uloskirjaudutaan, on sijoitettu oikeaan yläkulmaan, koska käyttäjillä on vakiintunut mielikuva siitä, missä painike sijaitsee. Tämä mielikuva on tullut, kun käyttäjät ovat käyttäneet muita sivuja, joissa uloskirjautuminen sijaitsee oikeassa yläkulmassa, kuten esimerkiksi facebook.com, google.com ja hs.fi sivustoilla. Ylärivin alla sijaitsee täytettävä lomake. Koska lomakkeessa on monta sivua eri huoneille, lomakkeen pitää olla eroteltu eri sivuille, koska muuten yhdelle ruudulle tulisi niin paljon tietoa, että käyttäjän olisi vaikea ymmärtää, missä kohdassa hän on menossa. Aktiivisena oleva sivu on merkitty eri värillä kuin muut sivut, joten käyttäjä muistaa helposti, missä kohdassa lomaketta hän on menossa. Alalaidassa olevasta painikkeesta (kohta 5 kuvassa), käyttäjä siirtyy seuraavalle sivulle. Viimeisellä sivulla painike vaihtuu lähetä-painikkeeksi, jonka avulla lomake lähettää tiedot palvelimelle.

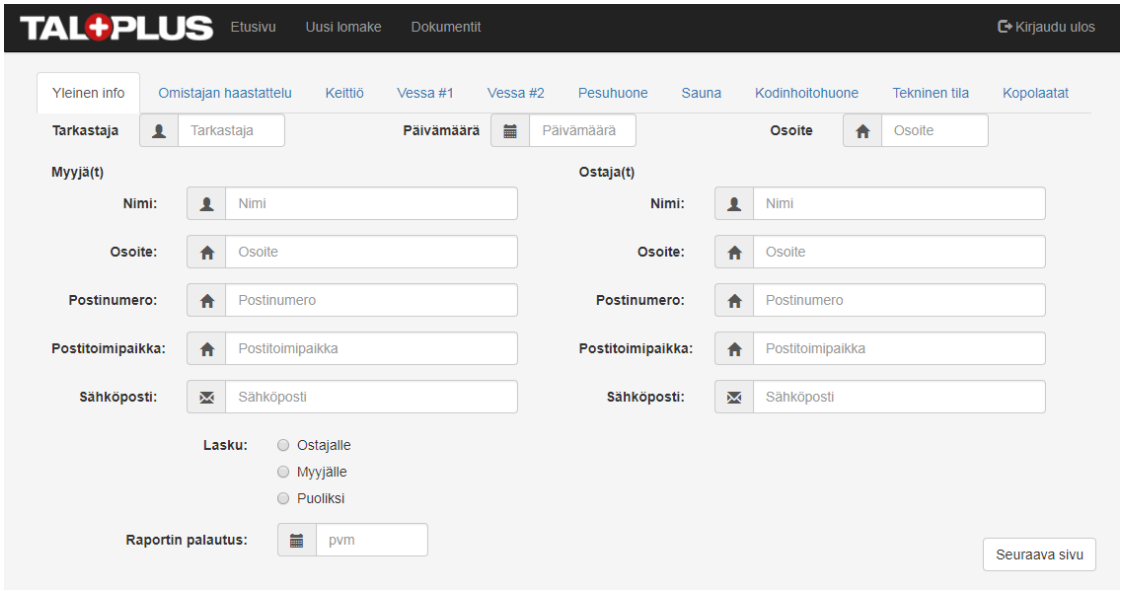
3.4 Sovelluksen toteutus

Aloitin sovelluksen ohjelmoinnin tekemällä ensin alustavan käyttöliittymän, joka perustuu suunnittelupiirroksiin. Käytin käyttöliittymän tekemiseen HTML form -rakenteita ja bootstrap-kirjastoa. Bootstrapin avulla voin tehdä käyttöliittymästä helposti skaalautuvan monelle eri päätelaitteelle, kuten kännykälle, tabletille ja laajakuvanäytölle.



KUVIO 9. Sovelluksen etusivu

Kuviossa 9 on sovelluksen etusivu. Tälle sivulle käyttäjä siirretään, kun hän on ensin kirjautunut sisään. Etusivun tarkoitus on olla siirtymäpiste muille sivuille ja samalla tarjota ajankohtaista tietoa käyttäjälle. Navigaatio-palkki on samannäköinen jokaisella sivulla, ja sitä kautta voi helposti siirtyä sivulta toiselle.



KUVIO 10. Kuva lomakkeen käyttöliittymästä

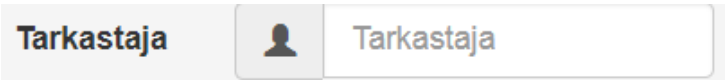
Kuviossa 10 on sovelluksen täytettävän lomakkeen ensimmäinen sivu, jonka kosteustarkastaja täyttää työtä tehdessä. Ensimmäisellä sivulla tarkastaja täyttää myyjän ja ostajan tiedot ja muuta yleistä tietoa kohteesta. Sivulla on enimmäkseen vain tekstikenttiä. Lomakkeen rakennetta ei ole muutettu, koska yritys halusi pitää sen samanlaisena alkuperäiseen paperiseen lomakkeeseen verrattuna.

Lomake perustuu erilaisiin HTML-elementteihin, joita ovat esimerkiksi text, checkbox, radio ja number. Text-elementteihin käyttäjä voi kirjoittaa tekstiä, ja se eroaa number-elementistä, siten että number-elementteihin käyttäjä voi syöttää ainoastaan numeroita. Näille elementeille bootstrap on valmiiksi määritellyt tyylin, joten niitä ei tarvinnut erikseen luoda.

```
<div class="input-group">
    <span class="input-group-addon"><i class="glyphicon glyphicon-user"></i></span>
    <input name="reportInspector" placeholder="Tarkastaja" class="form-control" type="text">
</div>
```

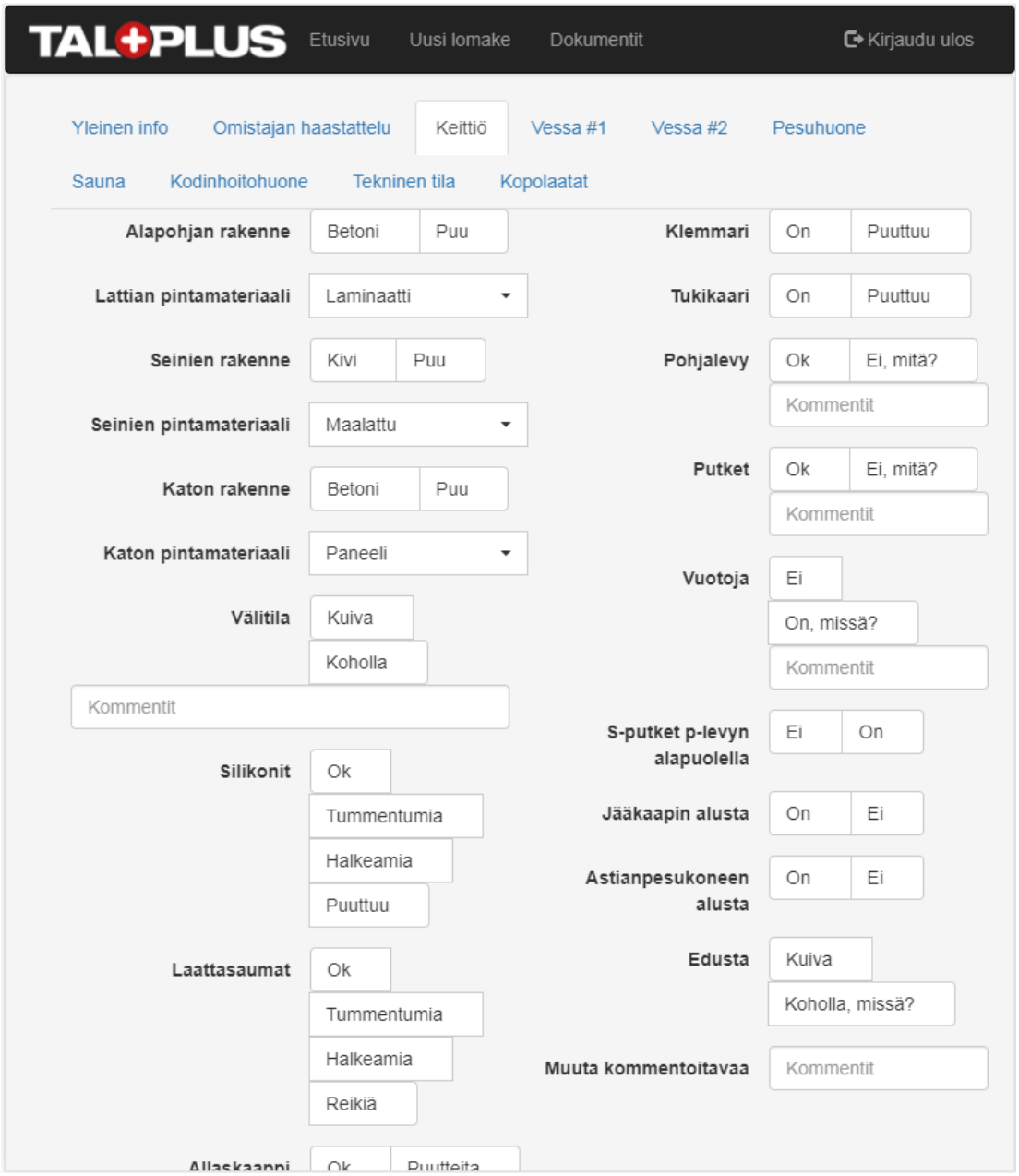
KUVIO 11. Katkelma lomakkeen koodista

Kuviossa 11 on koodipätkä etusivun kohdasta, johon tarkastaja laittaa nimensä. Se rakentuu bootstrap -kirjaston ”input-group” -luokkaan, joka luo tyylin elementeille. Kuviossa 12 on esitelty kuvion 11 koodipätkä, siten kuin se näkyy käyttäjän selaimella.



KUVIO 12. Kuvio 8 graafisesti näkyvänä käyttäjälle

HTML -elementti ”input” tarvitsee muuttujan ”type”, jotta oikeanlainen elementin tyyppi tulee valituksi, ja muutujan ”name”, jotta elementti voidaan tunnistaa myöhemmin. Siinä on lisäksi muuttujat placeholder ja class. Placeholder-muuttujan avulla kentässä on valmiiksi tekstiä käytettävyyden helpotukseksi, tässä tapauksessa teksti ”tarkastaja”, jotta käyttäjä tunnistaa helpommin, mitä hänen pitää kirjoittaa kenttään. Class muuttujan avulla elementti voidaan muotoilla näyttävämmäksi ja sitä voidaan käyttää myös elementin tunnistamiseen. Kuvassa olevan ””- elementin avulla luodaan kuvake kirjoituskentän vasemmalle puolelle, jotta koko elementti on selkeämpi käyttäjälle.



KUVIO 13. Lomakkeen keittiö -sivu

Kun tarkastaja on käynyt kohteen läpi ja täyttänyt lomakkeen kohdat, hän seuraavaksi painaa lähetä-painiketta. Esimerkki lomakkeesta on esitetty kuviossa 13. Lähetä-painiketta painamalla sovellus lähettää komennon palvelimelle, joka muuntaa saadut lomakkeen tiedot luettavaan muotoon, luo siitä Word-lomakkeen ja tallentaa sen tiedostona palvelimelle. Palvelimella toimiva koodi hakee ensin lähetetyt tiedot ja tallentaa ne yksittäisiin muuttujiin. Näiden muuttujien avulla oikeat tiedot tulevat oikeille sivuille, kuten esimerkiksi lomakkeen Keittiö-välilehden tiedot tulevat Word-dokumentin Keittiö sivuille.

```
75  if (isset($_POST['kitchenAlaRakenneCheckBox'])) {
76    $sectionKitchenData .= filter_input(INPUT_POST, 'kitchenAlaRakenneCheckBox', FILTER_SANITIZE_FULL_SPECIAL_CHARS);
77  }
```

KUVIO 14. Lomakkeen tiedon hakeminen

Kuviossa 14 on osa koodia, jonka toimintona on poimia Keittiö-sivulta alapohjan rakenne ja asettaa sen valittu arvo oikeaan muuttujaan. Tämä tieto on sovelluksen toiminnalle tärkeää, jotta oikea tieto siirtyy lomakkeelta raportille. Sovellus ensin tarkistaa sen, onko valintaruutuun syötetty tietoa ja, mikäli siinä on tietoa, se lukee tämän ja asettaa sen muuttujaan. Tätä tietoa hyödyntäen PHPWord saa tarvittavat tiedot raportin kokoamiseen.

```
153  // Adding kitchen section
154  $sectionKitchen = $phpWord->addSection();
155  $sectionKitchen->addText("2. HAVAINNOT KOHTEESTA JA TOIMENPIDESUOSITUKSET");
156  $sectionKitchen->addText("2.1. Keittiö");
157  $sectionKitchen->addText($sectionKitchenData);
158
```

KUVIO 15. Raportin luominen PHPWordin avulla

Kun PHP on kerännyt kaikki tarvittavat tiedot muuttujaan, seuraavaksi tiedot annetaan PHPWordille, jonka avulla oikea tieto lisätään oikeaan kohtaan raporttia. Kuviossa 15 on koodia, jossa PHPWord asettaa raportille otsikot ja niiden alapuolelle lomakkeelta kerätyt tiedot. Lomakkeelle asetetut tiedot kirjoitetaan raportille, ja ne luovat valmiiksi lauseita. Tässä tapauksessa lause olisi ”Alapohja on betonirakenteinen” tai ”Alapohja on puurakenteinen” riippuen siitä, kumman vaihtoehdon käyttäjä on valinnut.

```
205    $section->setFontStyle($frontpagefontStyle);
206    // Creating the header
207    $header = $section->addHeader();
208    $header->addImage('../img/logo2.png');
209
210    // Table for header
211    $headerTable = $header->addTable('myTable');
212    $headerTable->addRow();
213    $headerTable->addCell(5000)->addText("PINTAKOSTEUSMITTAUS");
214    $headerTable->addCell(5000)->addPreserveText('{PAGE}',$fontStyle, array('align' => 'right'));
```

KUVIO 16. Raportin ylätunnisteen luominen

Kuviossa 16 esitetään, kuinka PHPWord luo raportille ylätunnisteen. Yritys käyttää raporteissaan logoa ja sivunumerointia. Addheader-funktion avulla alustetaan kappaleelle tyhjä ylätunniste ja seuraavaksi addImagen avulla lisätään yrityksen logo. Seuraavaksi tunnisteeseen lisätään taulukon avulla otsikko ja sivunumerointi. Näin sivunumerointi saadaan aseteltua sivun oikeaan kulmaan.

4 POHDINTA

Tämän opinnäytetyön päätavoitteena oli luoda TaloPlus-yritykselle kosteustarkastukset digitalisoiva sovellus. Sovelluksen suunnittelu onnistui hyvin, ja yritys on tyytyväinen sovelluksen ulkoasuun ja tyyliin. Yritys myös pitää siitä, että lomakkeen rakenne pysyi samana kuin alkuperäinen paperinen lomake. Omasta mielestäni sovelluksen toteutus onnistui hyvin. Ohjelmointi- ja suunnittelutaitoni kehittyivät merkittävästi, ja pystyn nyt ajattelemaan paremmin kokonaisuutta enkä vain ohjelman tiettyjä ominaisuuksia.

Ensimmäisissä yritystapaamisissa sovimme, että sovellus olisi valmis kesällä 2017. Tämä oli mielestäni hyvä takaraja ja ehtisin siihen hyvin, sillä olin arvioinut suunnittelun ja toteutuksen vievän vähemmän aikaa. Lopulta aikaa kului toteutukseen suunniteltua enemmän ja jouduimme siirtämään käyttöönottoajankohtaa syksylle 2017. Huomasin aikatauluttamisen olevan luultua haastavampaa ja jouduin arvioimaan ajankäyttöni projektissa uudestaan.

Koska sain tehdä projektin alusta loppuun itsenäisesti, sain mahdollisuuden suunnitella ja aikatauluttaa sovelluksen tekemisen itse. Suunnittelussa otin huomioon yrityksen toiveet ja vaatimukset. Halusin tehdä käyttöliittymästä mahdollisimman yksinkertaisen, jotta yrityksen työntekijät oppisivat sen käytön mahdollisimman helposti ja nopeasti, jotta sovelluksen käyttö olisi sujuvaa yhdessä asiakkaiden kanssa. Päätin muuntaa lomakkeen sähköiseen muotoon, koska yrityksen työntekijät olivat tottuneet käyttämään sitä ja toivoivat, ettei sen rakennetta muutettaisi.

Sovellusta tehtäessä huomasin sovelluksen vaativan ominaisuuksia, joita en osannut suunnitteluvaiheessa ottaa huomioon. Suunnitteluvaiheessa en ollut ajatellut sovelluksen vaativan käyttäjienhallintaa. Käyttäjienhallinnalla tarkoitetaan sitä, että sovellukseen voidaan lisätä uusia sekä poistaa vanhoja käyttäjiä. Käyttäjien tietojen muokkaaminen on myös osa käyttäjienhallintaa. Käyttäjienhallinnan lisääminen sovellukseen lisäsi sovelluksen haastavuutta ja monimutkaisuutta.

Sovellus vaatii vielä hienosäätöä, kehittämistyötä ja testaamista todellisessa käytössä. Testaamisen avulla voidaan vahvistaa, että sovellus tuo aidosti arvoa yritykselle. Visuaalinen tyyli ja helppokäyttöisyys ovat osa-alueet, jotka myös tarvitsevat jatkokehitystä. Suosittelen, että yritys jatkaa sovelluksen kehittämistyötä, koska mielestäni sovelluksesta olisi hyötyä yritykselle, sillä sovellus vähentää työn määrää ja sitä voidaan laajentaa myöhemmin lisäominaisuuksilla.

Opin työn aikana kehittämään tänä päivänä todella tärkeitä työelämässä tarvittavia ”soft skillsejä” eli pehmeitä taitoja. Erityisesti sain tilaisuuden olla itse yhteydessä yritykseen ja sopia yritystapaamiset. Näiden tapaamisten ja yhteydenpidon avulla opin työelämässä tarvittavia taitoja. Koska yrityksen jäsenillä ei ole taustaa ohjelmistoista, jouduin miettimään, miten selitän asiat ymmärrettävästi. Koin tämän olevan kehittävää, sillä tulen tarvitsemaan tätä taitoa todella paljon työelämässä.

Opinnäytetyötä aloittaessa olin hyvin kiinnostunut aiheesta ja innoissani, että saan toteuttaa ohjelmiston kehittämisen alusta loppuun, sillä toivoisin pääseväni tekemään tälläisia tehtäviä työelämässä. Sovelluskehitys on ala, jossa sovellukset eivät ole koskaan täysin valmiita ja jossa niihin kehitetään jatkuvasti lisää ominaisuuksia, jotka vaativat ylläpitoa. Työn edetessä ominaisuuksien ja työn määrä alkoi kasvamaan, mikä johti siihen, että jouduin vetämään rajan, mihin asti toteutan sovellusta.

LÄHTEET

Aas, P., Dixit, S., Eden, S., Lawson, B., Moon, S., Wu, X. & O'Hara, S. 2018. HTML 5.3. Viitattu 25.2.2018, https://w3c.github.io/html/.

Achour M., Betz F., Dovgal A. & Lopes N. 2018. PHP Manual. Viitattu 22.3.2018

Addison Wesley Longman. 1998. Raggett on HTML 4 1998. Hakupäivä 10.1.2018, https://www.w3.org/People/Raggett/book4/ch02.html.

Bootstrap 2018a. Bootstrap. Viitattu 25.2.2018, https://getbootstrap.com/.

Bootstrap 2018b. Bootstrap. Viitattu 1.3.2018, https://getbootstrap.com/docs/3.3/css/.

Bootstrap 2018c. Bootstrap. Viitattu 22.3.2018, https://getbootstrap.com/docs/4.0/components/jumbotron/.

Gonzalez, J. 2013. Mobile first design with html5 and css3. Viitattu 22.3.2018

Håkon, W. L. & Bert, B. 1999. Cascading Style Sheets: Designing for the Web. 2nd edition Addison Wesley

Håkon, W. L. & Bert, B. 2005. Cascading Style Sheets: Designing for the Web. Viitattu 22.3.2018

Meloni, J., & Premier, D. S. 2003. Php essentials. Course Technology. Viitattu 25.2.2018

Niska, C. 2014. Extending Bootstrap. Birmingham: Packt Publishing.

PHPWord 2017a. Introduction. Viitattu 15.3.2018, https://phpword.readthedocs.io/en/latest/intro.html.

PHPWord 2017b. General usage. Viitattu 21.3.2018, https://phpword.readthedocs.io/en/latest/general.html.

Pouncey, I. & York, R. 2011. Beginning CSS: Cascading Style Sheets for Web Design. Indianapolis, Ind: Wrox.

Suomen virallinen tilasto (SVT) 2017. Väestön tieto- ja viestintätekniikan käyttö: 2. Internetin käyttö mobiililaitteilla. Viitattu 22.3.2018, http://www.stat.fi/til/sutivi/2017/13/sutivi_2017_13_2017-11-22_kat_002_fi.html.

Taloplus 2018. Tervetuloa tutustumaan Taloplus Oy:n tarjoamiin palveluihin!. Viitattu 26.3.2017, http://taloplus.fi/.

Tanninen, J. 2017. Toimitusjohtaja, Taloplus Oy. Haastattelu 18.10.2016

Työ- ja elinkeinoministeriö. 2015. Palvelutalouden murros ja digitalisaatio - Suomen kasvun mahdollisuudet. Viitattu 22.3.2018, http://julkaisut.valtioneuvosto.fi/bitstream/handle/10024/74984/TEMjul_12_2015_web_30032015.pdf?sequence=1&isAllowed=y.